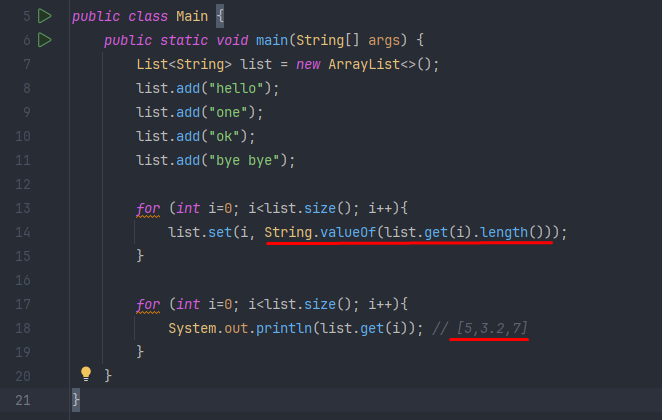
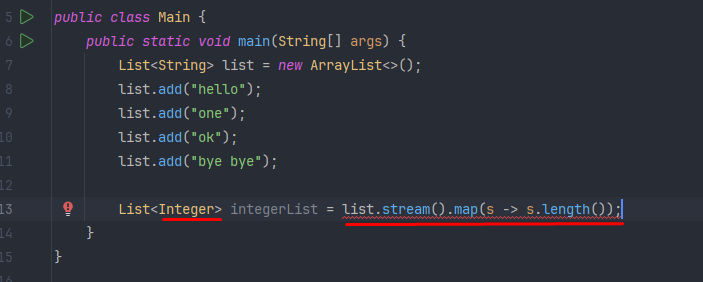


Stream elementlarni o’zida saqlamaydi. Stream shunchaki elementlarni Array, Collection yoki IO chanel(File) dan oladi va ular ustida amal bajaradi. Stream hech qanday sourceni o’zgartirmaydi, ya’ni asl Arrayni yoki asl Collectionni o’zgartirmaydi. Stream shu Array yoki Collection ni har bir elementini yurib chiqadi. Shu yurib chiqish davomida qanaqadir operatsiya bajaradi.

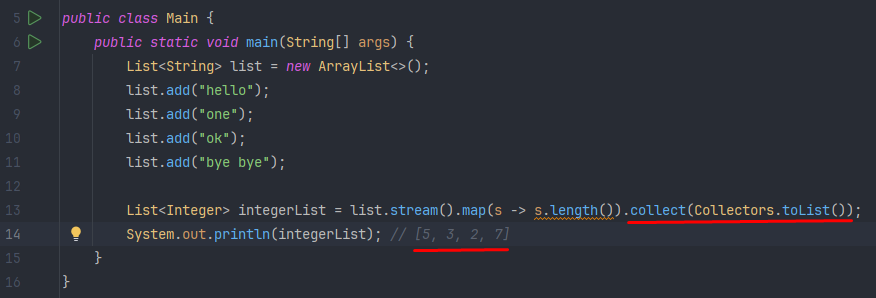
Keling streamni ko’rishdan oldin, bir misol ko’raylik. Deylik bizda **String** typeli **AL** bor. Bizga shu listda shu listdagi stringni uzunligini joylashtirib chiqish so’raldi. Bunday vaziyatda biz bunday qilamiz:



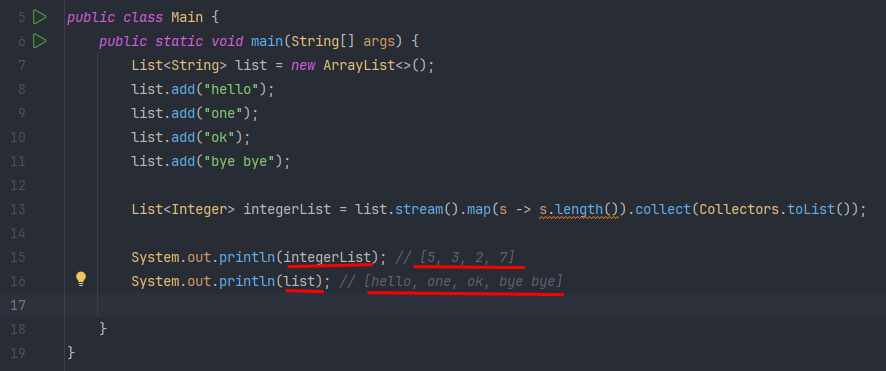
Lekin bu ishni stream bilan ham osonroq hal qilish mumkin. Pastdagi misolda Listni stream() methodi orqali streamga o’zgartiryapmiz. E’tibor bergan bo’lsangiz xatolik beryapti. Sababi pastdagi map() method bizga stream qaytaradi, biz esa shu streamni List<Integer> ga o’girmoqchi bo’lyapmiz, ta’biyki xatolik beradi:



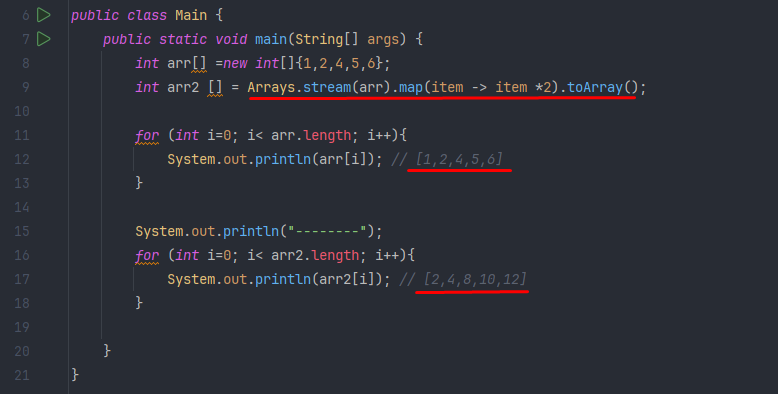
Bu xatolikdan qutulish uchun bizga collect() methodi kerak bo’ladi. Bu method stream ni list ga o’giradi:



Birorta stream methodlari asl collection(List, ArrayList, LinkedList,…)ni yoki massivni o’zgartirmaydi. Pastda biz lsit ni stream ga o’girganimiz bilan, asl listni ko’radigan bo’lsak, asl list o’zgarmaganligini ko’rishimiz mumkin. Oldin qanday qiymatlar bo’lsa, hozir ham shu qiymatlar mavjud:



Xuddi shunday massivlarni ham streamga o’girib, shu stream ustida amal bajarib, yana uni massivga o’girishimiz mumkin:

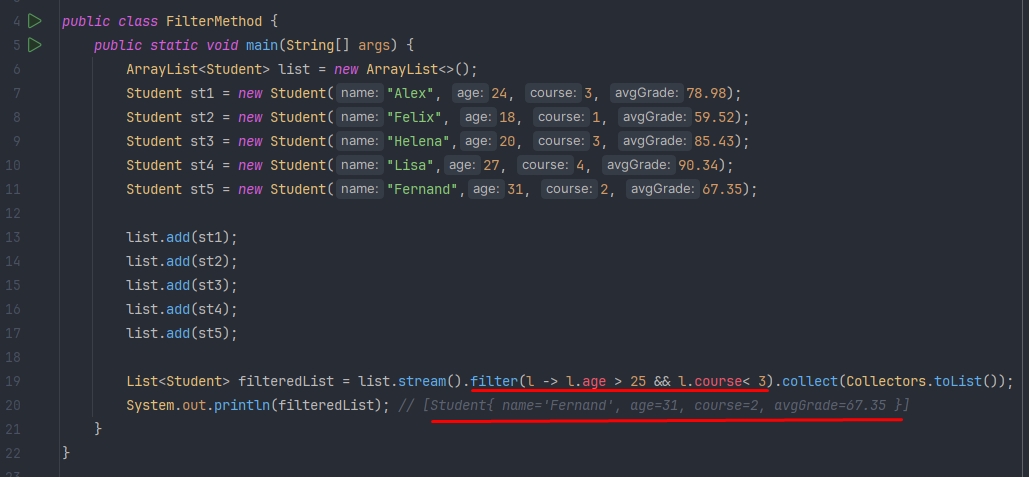


**Filter()**

Bu method JS dagi filter() method bilan bir xildir, ya’ni qanaqadir shartga tushadiganlarini chiqarib beradi. Pastda Student class bor:

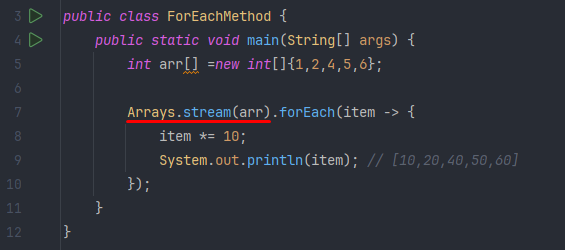


Pastda esa filter method yozganmiz, bu method ham stream qaytaradi va uni yana Listga o’girishimiz kerak. filter() m-dni ichidagi shartga ko’ra, age > 25 va course<3 dan kichiklarini chiqar deyapmiz. Faqat bitta item shu shartni qanoatlantirar ekan:



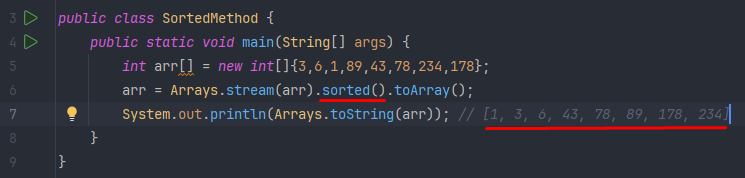
**ForEach()**

Bu method hech qanday qiymat qaytarmaydi(void typelidir), xattoki stream ham qaytarmaydi. Bu method asosan print qilish uchun ishlatiladi:

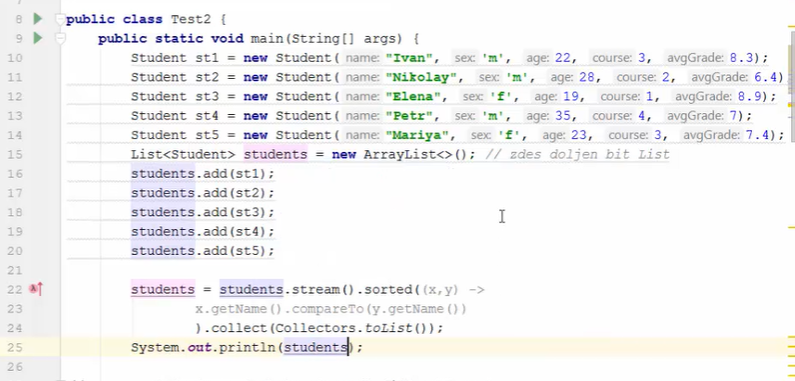


**Sorted()**

Sorted() method sort qilish uchun ishlatiladi:

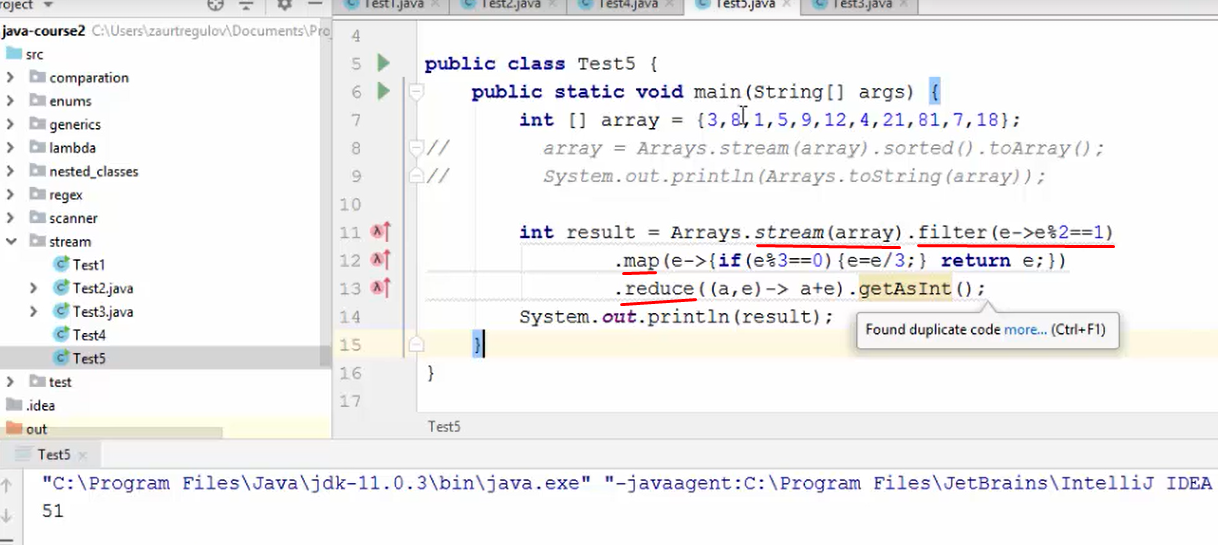


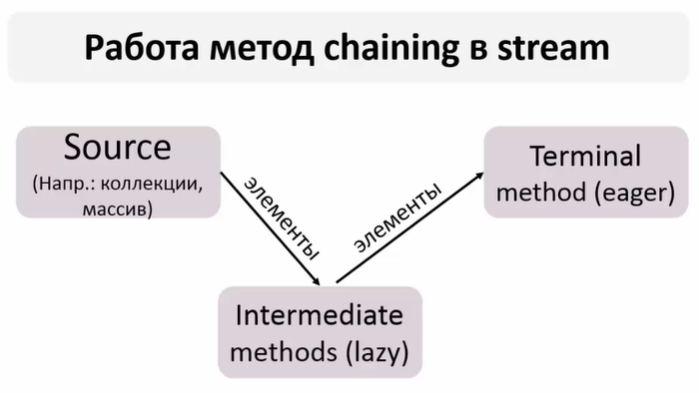
Istasak Student objectida biror bir field bo’yicha sort qilish mumkin:



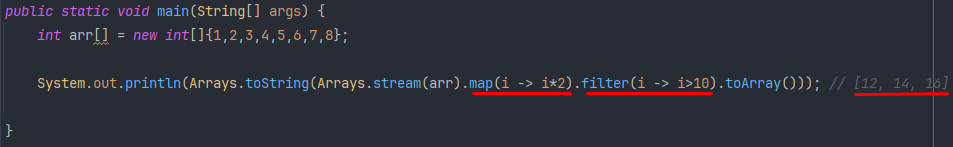
**Method chaining**

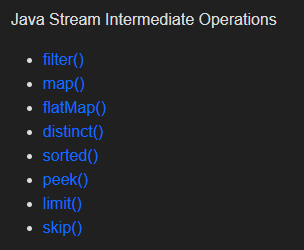
Bunda bir nechta stream methodlarini ketma-ket ishlatish mumkin:



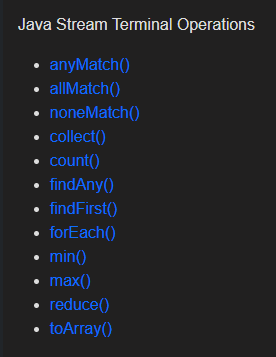


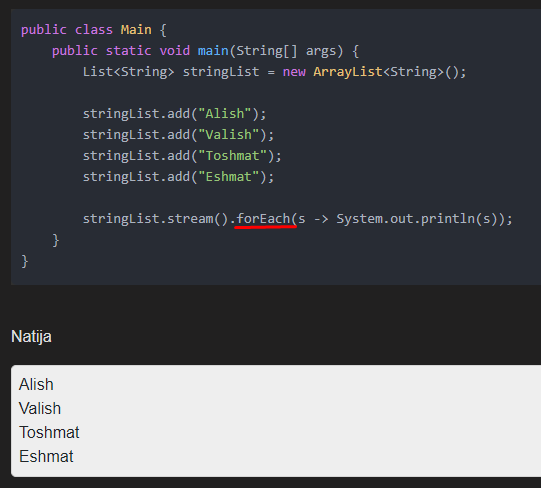
**Intermediate methodlar(lazy) -** bu stream ni olib yana stream qaytaradi, masalan stream olib uni ustida qanaqadir amal bajarib yana stream qaytaradi. Deylik bizda int typeli array bor bo’lib, bu array arrayni map method orqali elementlarni 2 ga ko’paytiramiz. Bu method stream ni qaytaradi. Endi shu qaytgan streamdan biz filter methodni ishlatib 10 dan kattalarini chiqaryapmiz. O’z navbatida bu filter method ham stream qaytaradi.





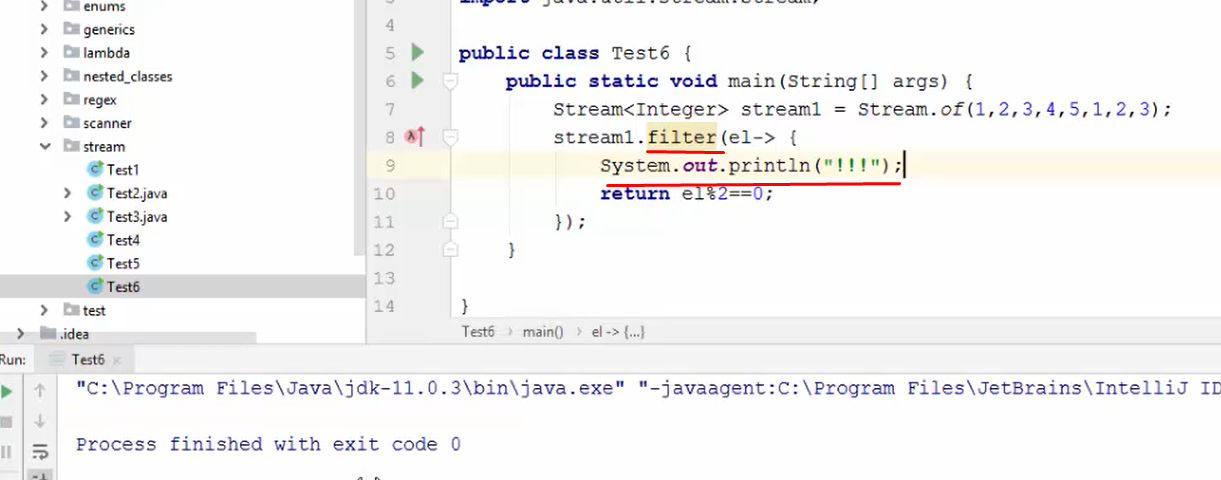
**Terminal methodlar(eager)** – bu methodlar stream qaytarmaydi. Bu methodlar eng oxirida ishlatiladi, ya’ni intermediate methodlardan qaytgan stream ustida ish bajaradi:

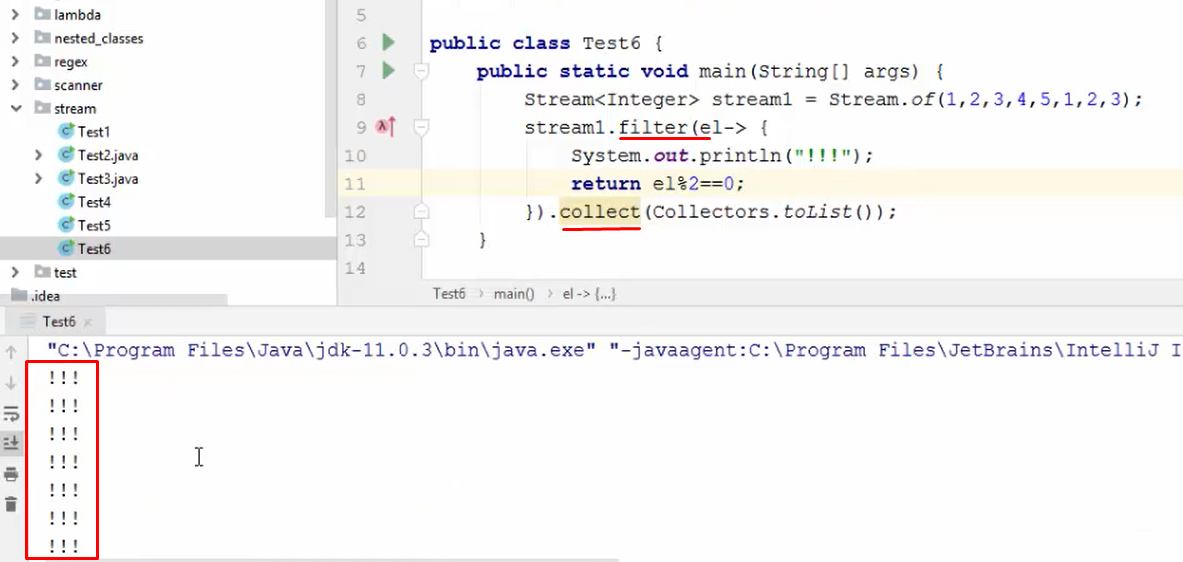




Bir qiziq holat bor bo’lib, intermediate methodlar bajarilmaydi to terminal methodlar bajarilmaguncha.

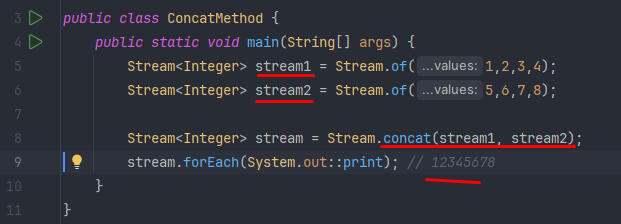
Keling shu narsani isbotlab ko’ramiz. Masalan pastda intermediate method(**filter()**)ni ishlatganmiz va bu methodni ichida console yozganmiz. Lekin ishga tushurib ko’rsak, hech qanday console chiqmaydi:



Lekin biz intermediate methoddan keyin terminal method(**collect()**)ni ishlatsak, u holda bizga console ni chiqaradi. Demak intermediate methodni ishlashi uchun terminal methodni yozishimiz shart ekan:

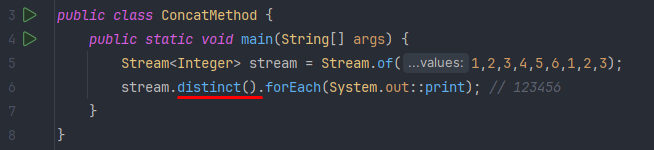
**Concat()**

Concat methodni ishlatilishi:



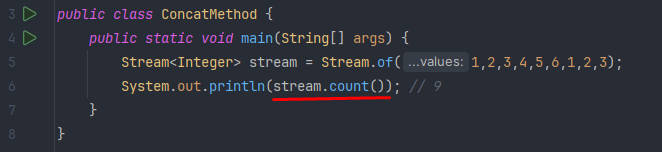
**Distinct()**

Distinct() method stream da takrorlanganlarini chiqarmaydi:

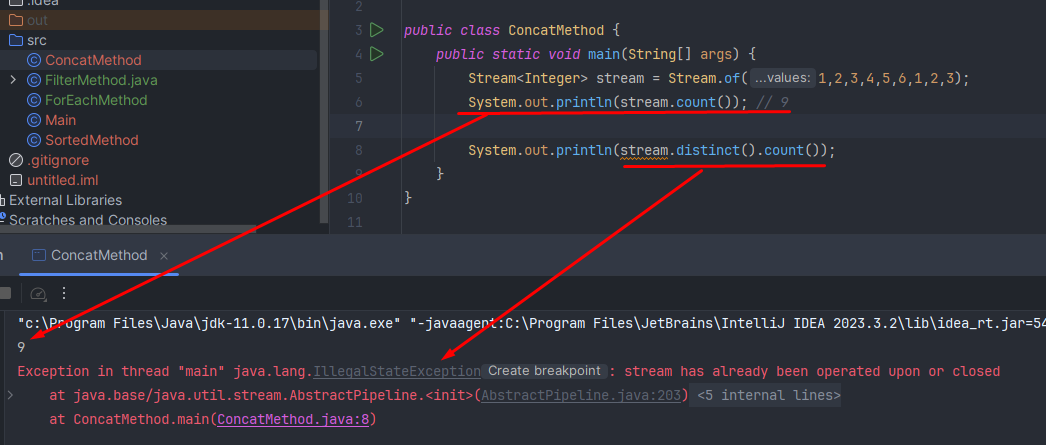


**Count()**

Count() method bizga streamdagi elementlarni sonini qaytaradi:

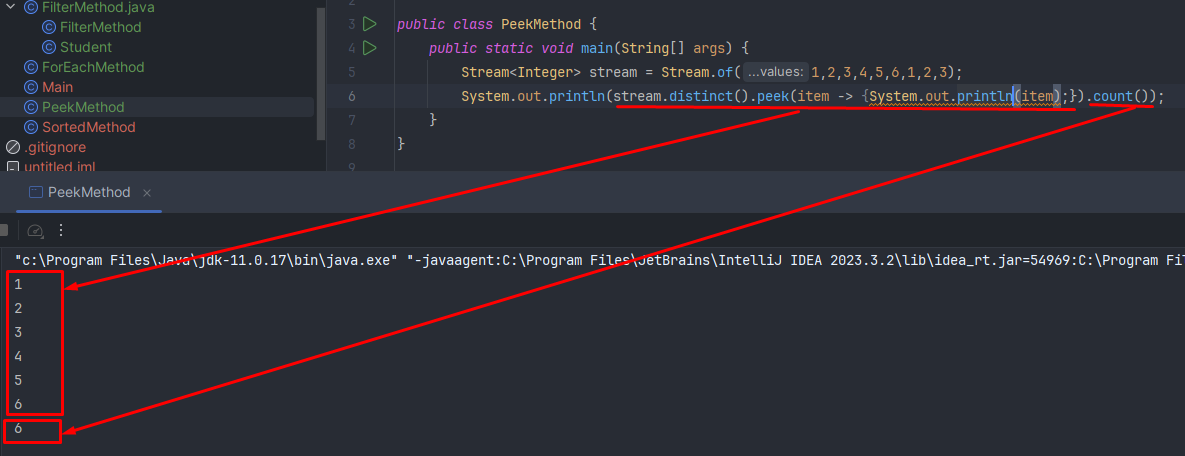


Stream larda agar biz biror streamni ustida amal bajargan bo’lsak, va uni ustida yana qaytadan amal bajarmoqchi bo’lsak, u holda bizga pastdagi exception tashlanadi. Sababi 6-qatorda streamni count() methodini ishlatdik, lekin 8-qatorda yana shu stream ustida amal bajarmoqchi bo’lyapmiz, bu esa exception ga sabab bo’lar ekan. Demak streamlarda bir stream ustida 2 va undan ortiq amal bajarib bo’lmas ekan:



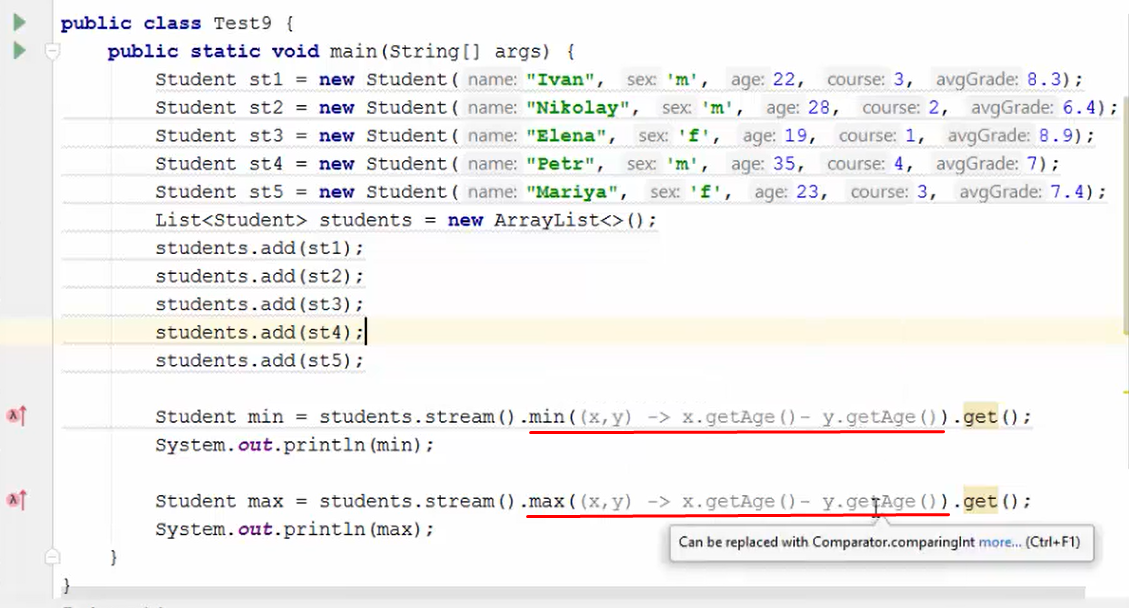
**Peek()**

Bu method foreach() method bilan bir xil ishlaydi, asosiy farqi foreach bizga hech qanday stream qaytarmas xattoki hech qanday qiymat qaytarmas edi. Lekin bu method bizga stream qaytaradi. Pastda ko’rish mumkinki, peek() methoddan qaytgan streamni count ini olyapmiz:



**Min and Max**

Eng kichik va eng katta elementlarni streamdan oladigan methoddir. Pastda min va max ni age bo’yicha topib olyapmiz:



**Parallel stream**

